ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

STUDY OF DEFECT, ERRORS AND TESTING CHALLENGES IN WEBSITE USING MANUAL AND AUTOMATION TESTING TECHNIQUES

Bharti Bhattad*, Prof. Abhay kothari
* Acropolis institute of technology and research Dewas Bypass Road, Indore, India
Acropolis Institute of Technology and research Dewas Bypass Road, Indore, India

ABSTRACT

Testing is a quality assurance of the application development; therefore it is must essential for new products. In this paper a software quality scheme is introduced for web based application testing. The proposed testing framework includes the stress testing, load testing, performance testing, and unit testing of the web projects. The proposed testing tool is a set of testing methods by which a web application tested on different performance parameters. Additionally the implementation of the proposed technique is also reported using visual studio environment. The performance of the reported system is estimated in terms of server response time in terms of time and number of users. Also calculate the time and space complexity respectively. It found the adoptable and efficient testing support for the intermediate module testing of web applications.

KEYWORDS: software testing, web application testing, unit testing, stress testing, server response time .

INTRODUCTION

We can evaluate Quality Assurance of any product by testing. Thus for a quality of product development needs an appropriate testing strategies. In this study web, application tested in order to producing an efficient and error free web application. Now in these days the world becomes more competitive additionally service oriented business are growing rapidly, in order to provide the continuous services to the end clients. Advance and new technologies employed for customer reliability and satisfaction. This may become more open and convenient with online communication, shopping, banking and other applications. Therefore, most of the real life works becomes online services, which provide the 24X7 connectivity with clients. For that, purpose a good quality of application is desired which provide the services continuously during different conditions. Software engineering provides techniques for application development and design with quality assurance. The testing of applications performs the measurement of quality in software engineering. Therefore testing is a crucial and essential for the good quality application development.

In this thesis work, testing techniques and their strategies are discuss and investigated. More specifically automated web based software-testing techniques for finding the efficient and supportive technique for application development and deployment. Therefore, a detailed study on web application architecture and their development and deployment environment is studied. In addition, of that, a new web application-testing tool proposed for implementation. The proposed web application-testing tool incorporates different modules and methodologies for producing the enhanced and quality web application.

This section provides a general overview of the presented testing tool, the key goals and objectives of the presented work to be discussing in further sections.

BACKGROUND

Software testing can be understands as the process of validating and verifying that a software program/application/product [1]:

1. Meet the requirements that guided its design and development;
2. Works as expected;

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Can be implement with the same characteristics. In a more traditional software development model, most of the test execution occurs after the requirements have been refined and the coding process has been completed [2]. Each aspect described in the previous list produces new testing challenges and perspectives. For example, access concurrently by a large number of users. Moreover, as users may utilises browsers with different Web content rendering capabilities [3], Web applications must be test to make sure that the expected application's behaviour using different Web browsers, operating systems, and middle ware is the one expected. As for the existence of dynamically generated software components, the issue here is to cope with the difficulty of generating and rerunning the same conditions that produced each component [4].

The remainder of this chapter uses the term Web application (or simply application) to indicate the set of software components implementing the functionality and services the application provides to its users, while the term running environment will indicate the whole infrastructure (composed of hardware, software and middle ware components)needed to execute a Web application [5]. The main goal of testing a Web application is to run the application using combinations of input and state to discover failures. A failure is the manifested inability of a system or component to perform a required function within specified performance requirements [6]. Since a Web application is strictly inter woven to its running environment, it is not possible to test it separately to find out exactly what component is responsible for each exhibited failure. Therefore, different types of testing have to be executes to uncover these diverse types of failures [7]. Thus, Web application testing will be judge from two distinct perspectives.

One perspective identifies the different types of testing that need to be executing to verify the conformance of a Web application with specified non-functional requirements.

The other perspective considers the problem of testing the functional requirements of an application. It is necessary that an application be testing from both perspectives, since they are complementary and not mutually exclusive [8]. Usually, performance testing is executing by simulating thousands, or even more, parallel user accesses over a defined time interval [9].

As for the difficulties of executing load testing of Web applications, considerations similar to the ones made for performance testing can also be taking into account. Failures found in load testing are mainly due to faults in the running environment [10]. In the case of Web applications, stress-testing problems are similar to those that can be meeting in performance and load testing [11].

Usability is a critical issue for a Web application. Indeed, it may determine the success of the application [6]. The application is mainly responsible for usability failures. Accessibility testing is considered as a particular type of usability testing, whose aim is to verify the access to an application's content is allowed even in the presence of reduced hardware and software configurations on the client side (e.g. browser configurations disabling graphical visualisation, or scripting execution) [12], or in the presence of users with disabilities, such as visual impairment.

In the case of Web applications, accessibility rules such as the one provided by the Web Content Accessibility Guidelines [13] established, so that accessibility testing represents verification the compliance of an application with such rules.

Buchler et al [14] present SPaCiTE. This tool relies on a dedicated model-checker for security analyses that generates potential attacks with regard to common vulnerabilities in web applications. Yongpo Liu et al [15] present the design of generic codec for testing Web application.

Jason Bauet al [16] used a custom web application vulnerable to known and projected

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

vulnerabilities, and previous editions of widely used web applications containing known vulnerabilities.
Shauvik Roy Choudhary et al [17] presents X-PERT a tool for identifying XBIs in web applications automatically, without requiring any effort from the developer, X-PERT implements a comprehensive technique for identifying XBIs and has been shown to be effective in detecting real-world XBIs in empirical evaluation of the software. The source/object code of X-PERT and XBI reports from evaluation are available at link http://gatech.github.io/xpert

Nadia Alshahwan et al [18] introduced three related algorithms and a tool, SWAT, for automated web application testing using Search Based Software Testing (SBST). Maurizio Leotta et al [19] present an industrial case study about test automation and test suite maintenance in the context of Web applications.
Fathy E. Eassa et al [20] an integrated multi-agent testing tool presented. Such tool comprises static analyser, dynamic tester and an integrator of the two components for detecting security vulnerabilities and errors in agent based web applications written in Java.

Ali Mesbah et al [21] propose a method for testing AJAX applications automatically, based on a crawler to infer/use a state-flow graph for all (client-side) user interface states. We describe somehow three case studies, including six subjects, evaluating the type of invariants that can be obtained for AJAX applications as well as these fault revealing capabilities, scalability, required manual effort, and level of automation of given testing approach [22].

PROPOSED WORK

The proposed work intended to find a best testing strategy for web based application testing. Now in these days the traditional desktop applications are becomes online-based systems. Therefore, web application development is in on high demand. On the other hand, these applications are using to store sensitive and important data therefore the security and other issues are also involve in this domain. In addition of that the application development and their internal architecture is different from the traditional desktop-based application development. Thus, a different kind of strategy is required to test a web application. The testing of a web based application need the following objective to test with any complete web application-testing tool.
In order to implement the desired concept to test and support application development and deployment of web based software products a modular architecture suggested using the figure 3.1.

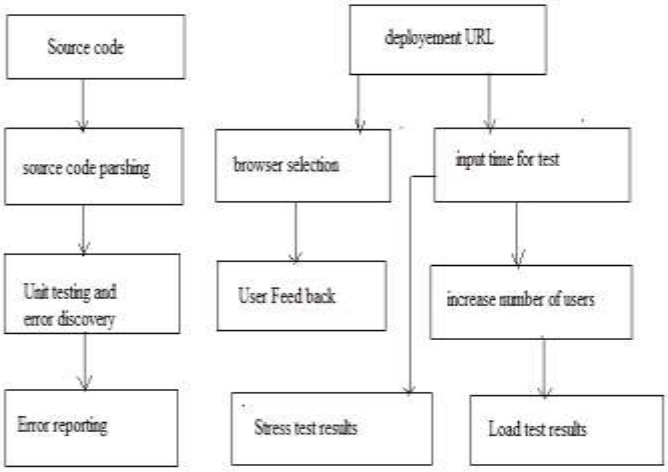


Figure 3.1: Proposed Testing Model

The proposed system is accepts two different input for perform testing on the web based application. All the system designed in modules and component basis; therefore, all the sub components and modules are discusses here in detail.

Source Code

The testing tool accepts the source code repository for finding bugs and error in the existing code. The web-based system contains more than one type of source code, such as java script, CSS and ASP.net source code files. For testing of java scripts and CSS files not any useful resources are available therefore only

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

ASP.net source code files are tested for the their bugs and errors.

Source Code Parsing
In this, module the input source code parsed first, which categorize the available source code for classes, methods, functions and their participating variables. In this step, the source code files separated and remaining support documents filtered. The filtered server side codes tested in next phase.

Unit testing and Error Discovery
In this phase, the ASP.NET source code files tested for their functions using randomly generated inputs. In this phase, a third party API used to produce and generate values for function. The resultant bugs and errors listed in this phase. On the other hand, the application tested using the deployment URL.

Error Reporting
The evaluation results of the Unit testing are giving in this part of the system.

Deployment URL
The application deployed in remote host or server machine, and for accessing the application, using client browser a domain name is associated with this deployed server. In addition, client access the documents and application modules using this domain name and using the path. This path or URL produced as input to the system for other kind of testing.

Browser Selection
Using the deployed URL the application is testing for their compatibility with the different web browsers, such as Fire-Fox from Mozilla, internet explorer, Chrome from Google, Opera and many more.

User Feedback
In this phase, the user feedback for compatibility issues and their performance is evaluated using users feedback. The input user feedback collected in terms of rating values between 0-10.

Input Time for Test
The performance of the web application tested for 24-hour performance therefore for evaluating the performance of web application the load on server at different time is required to evaluate. In addition, this will results the different time based responses of the server.

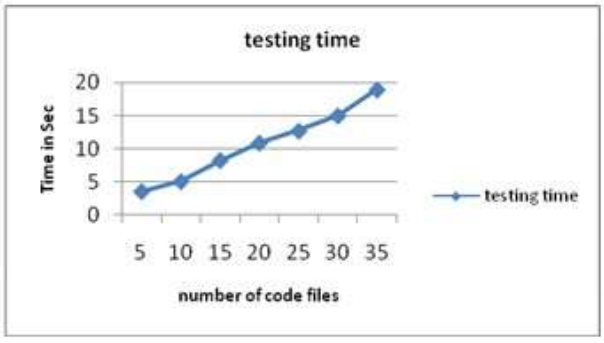
Increase Number of Users
On the other hand, the numbers of users are increases to find the performance of web application during increasing concurrent users to the system.

Load Testing Results
In this phase, the outcome of the system concluded and demonstrated using suitable performance parameters.
This phase discussed about the different testing modules incorporated with the system, in next section the summary of the chapter presented.

RESULTS ANALYSIS

The proposed testing tool is a set of testing methods by which a web application tested on different performance parameters. With the system, experimentation and different conditions during testing different outcomes evaluated and listed in this section.

Time Complexity
Time complexity also known as time consumption that estimated using the time difference between initialization of code processing and the finalizing of testing. In this context, the total time required to perform the test over web application's code files is termed as time complexity.



ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Figure 5.1 Unit testing results

Figure 5.1 shows the unit testing performance of proposed web application testing tool. In this diagram, the X-axis shows the amount of code files for testing and Y-axis shows the time consumed during testing. The web application consists of various other scripts and supporting files are included. Thus, these files are not included during the testing results demonstration. According to results, the amount of time required to test the code files. There are two facts found:

1. More number of code files leads more time consumption
2. If number of code lines in a code is larger than the amount of time required to test the application is also increases in the same manner.

Space Complexity

The amount of main memory required to test an application is termed here as time complexity of the system. Time complexities of the testing tools are considering in two different parts. First memory consumption during the load testing that is demonstrated using figure 5.2 and memory consumption during code testing which is demonstrated using figure 5.3.

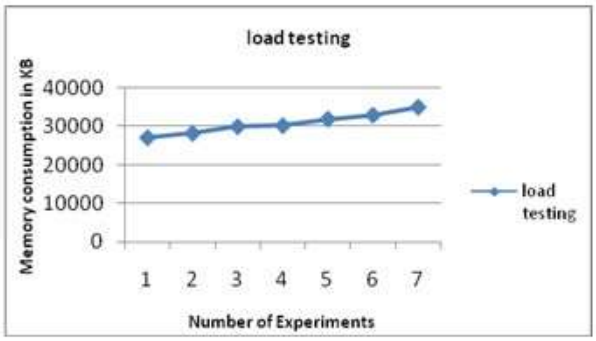


Figure 5.2 shows the space complexity during the load testing, during load, testing the client machine main memory consumption estimated and during each experiment 5 numbers of users are increased. During this testing environment, the estimated results are given using figure 5.2. According to the evaluated results as the number of users increase, the memory consumption of the testing tool is increases in the similar manner. In this diagram, the amount of main memory given using Y-axis and the X-axis shows the number of experiments.

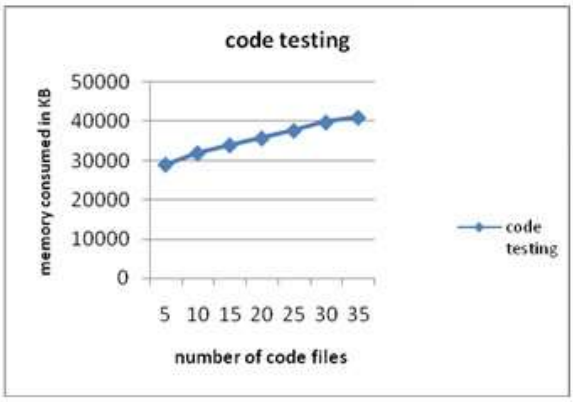


Figure 5.3: Space Complexities during Code Testing

The figure 5.3 shows the performance of system during code testing, in this context the amount of main memory consumed tested during increasing number of code files to test. Therefore, the amount of code files are represented using X-axis and the Y-axis demonstrates the amount of main memory consumed in terms of kilobytes. According to the obtained results the as the number of code files for testing is increases the amount of main memory consumption is increases in respective manner.

Response Time

The amount of time required to generate response by the remote host for a given request known as the server response time. During number of users in the system increases, the testing of server response time collected for 20 minutes and average response time of the server is calculated and demonstrated using figure 5.4. In this diagram, the X-axis shows the number of users in system and the Y-axis shows the amount of time consumed for providing the response to the end client. According to the results as the number of user, increases in a host the amount of time consumption are increases in the same manner.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

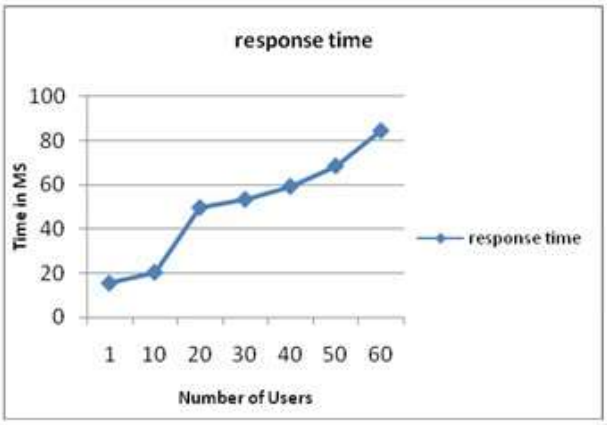


Figure 5.4: Server Response Time

ERROR DETECTION RATE

The amount of error detected during different experimentation with the source codes is termed as error detection rate. The estimated number of errors during different experiments is listed using figure 5.5. In this diagram, the amount of error detected given in Y-axis and the X-axis demonstrates the number of experiments performed with the system. According to the obtained results the amount of error detection is not depends upon the number of code files that is directly depends upon the source code and their development and deployment environment.

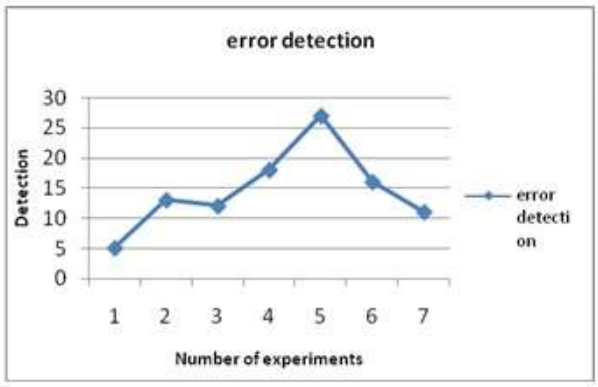


Figure 5.5 Error Detection Rate

CONCLUSION AND FUTURE WORK

The given section draws the conclusion of performed study, therefore during the development of web testing tool and during experimentation with system, essential facts are listed here. In addition, of that, the limitations and future extension are also listing in this section. The main objective of the online applications is to provide user specific services in 24X7 manners. There are various applications such as shopping web; banking and other CMS applications are available online. In order to provide continuous service to the end client, quality applications are required to develop. Therefore, in this work, a web application-testing tool is developed and designed. The proposed testing model incorporates different testing mechanism for delivering the quality product during different deployment phases.

The main features of the developed tool are web application functional testing, computability analysis with web browsers during deployment of the application. In addition, of that, during development using unit testing the application provides support to developer for enhancing the application functions and classes. The implementation of the proposed testing tool performed using visual studio environment. The facts of performance can be discussing as:

1. Time complexity of the system is low, but as the number of code file under evaluation is increased the amount of time, required compiling and test functions increased with the ratio of number of code files and line of codes.
2. Space complexity of the system is low, that is not depends upon the number of code file input. That is directly proportional to the amount of code line required to compile.
3. Response time of the web applications is increases as the load on server is increases, therefore more the number of concurrent users causes the similar amount of load. On the other hand, if the amount of response time increases than, it will not affect much on the server load.
4. Error detection rate is not depends upon the files or the code lines. It is

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

directly depends on the deployment environment and incorrectly defined code blocks.

In near future required to perform more research on different code environments and script validation methodologies. For extending the proposed testing tool for different other programming languages.

REFERENCES

1. N. Alshahwan and M. Harman, "Automated Web Application Testing Using SearchBased Software Engineering", 26th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2011.
2. F. E. Eassa, M. Zaki, A. M. Eassa and T. Aljehani, "IMATT: An Integrated Multi-Agent Testing Tool for the Security of Agent-Based Web Applications", World Journal of Computer Application and Technology vol 1, no 2, pp. 19-28, 2013.
3. A. R. Arunachalam, "Innovative Approach of Testing in Event Driven Software (EDS)", Indian Journal of Science and Technology, vol 7, no s5, pp. 37–40, June 2014.
4. A. Mesbah, A. Deursen and D. Roes, "Invariant-Based Automatic Testing of Modern Web Applications", IEEE Transactions on software engineering, vol. X, no. Y, 2011.
5. S. Elbaum, G. Rothermel, S. Karre, and M. Fisher, "Leveraging User-Session Data to Support Web Application Testing", IEEE Transactions on software engineering, vol. 31, no. 3, March 2005.
6. J. T. Yang, J. L. Huang, F. J. Wang, and William. C. Chu, "An Object Oriented Architecture Supporting Web Application Testing", 0-7695-0368-3/99 $10.00 0 1999 EEE.
7. A. Arora and M. Sinha, "Web Application Testing: A Review on Techniques, Tools and State of Art", International Journal of Scientific & Engineering Research, vol. 3, no. 2, ISSN 2229-5518, February-2012.
8. V. Dallmeier, M. Burger, T. Orth and A. Zeller, "Web Mate: A Tool for Testing Web 2.0 Applications", JSTools'12, Beijing, China, Copyright 2012 ACM 978-1-4503-1274-5/12/06, June 13, 2012.
9. Dr. R. Kumar P., and K. Bhargav, "A Survey on Performance Testing Approaches of Web Application and Importance of WAN Simulation in Performance Testing", International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 05, May 2012.
10. J. Gao, X. Bai, and W. T. Tsai, "Cloud Testing- Issues, Challenges, Needs and Practice", Software Engineering: An International Journal (SEIJ), vol. 1, no. 1, September 2011.
11. H. J. Kam and J. J. Pauli, "Work in Progress - Web Penetration Testing: Effectiveness of Student Learning in Web Application Security", 978-1-61284-469-5/11/$26.00 ©2011 IEEE.
12. N. Alshahwan, "Utilizing Output in Web Application Server-Side Testing", Department of Computer Science University College London, August 19, 2012.
13. J. Offutt, V. Papadimitriou, and U. Praphamontripong, "A Case Study on Bypass Testing of Web Applications", Accepted for publication, Empirical Software Engineering journal, 20-June-2012.
14. J. Conallen, "Building Web Applications with UML: Web Application Basics", available at http://www.informit.com/articles/article.aspx?p=30610 , Jan 17, 2003.
15. Available at http://www.pearsonhighered.com/samplechapter/0201730383.pdf
16. M. Leotta, D. Clerissi, F. Ricca and C. Spadaro, "Repairing Selenium Test Cases:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

An Industrial Case Study about Web Page Element Localization", © 2013 IEEE.

17. Y. Zou, C. Fang, Z. Chen, X. Zhang and Z. Zhao, "A Hybrid Coverage Criterion for Dynamic Web Testing", at http://software.nju.edu.cn/zychen/paper/2013SEKEa.pdf
18. S. Bucur, J. Kinder and G. Cande, "Making Automated Testing of Cloud Applications an Integral Component of PaaS", fourth Asia-Pacific Workshop on Systems (APSYS), Singapore, July 2013.
19. Z. Qian, "Towards Testing Web Applications Using Functional Components", Journals of software, vol. 6, no. 4, April 2011
20. F. Lazarinis, S. Green and E. Pearson, "Creating personalized assessments based on learner knowledge and objectives in a hypermedia Web testing application", 0360-1315/$, 2010
21. A. S. Brar and A. Jalota, "Implementation of an optimal approach to testing web based applications", Asian Journal Of Computer Science And Information Technology vol 3, no 2 , pp. 26 – 28, 2013
22. Available at http://www.w3schools.com/ajax/ajax_intro.asp.